Anglia Ruskin University Cambridge & Chelmsford · IEEE · University of L'Aquila · IET
Faculty of Electrical Engineering, University of Sarajevo

TRACK LAYOUT DESIGN USING MODIFIED GENETIC ALGORITHM

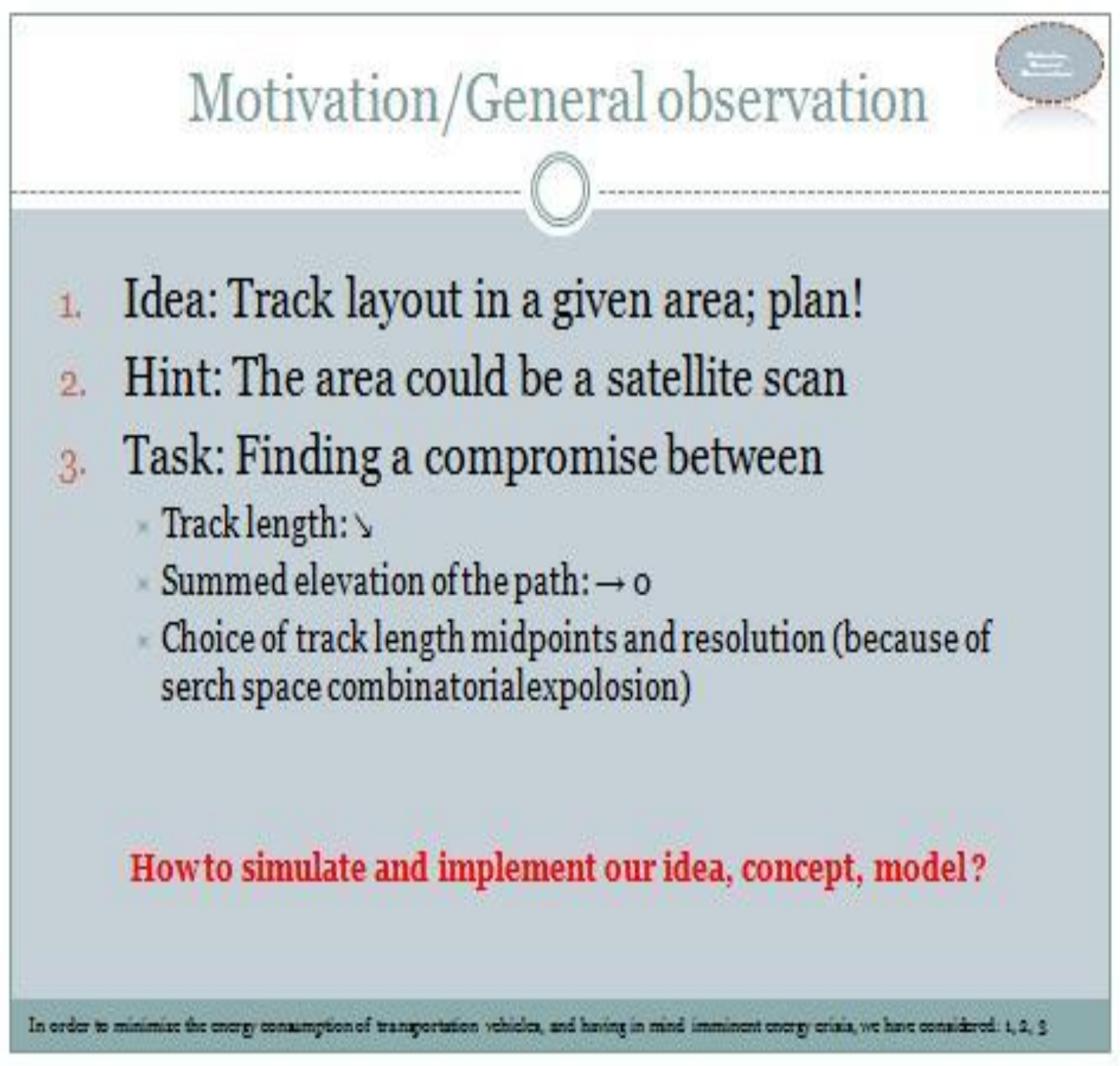
ISIE'08
Cambridge 3 rd, July

Z. Avdagić, A. Smajević, D.Bošković

Abstract **Track or road layout in a given geographical area to plan new or improve existing public and/or private transportation systems is a complex problem. Especially today, many parameters have to be considered, and some of them are not obvious. For example, in face of an imminent energy crisis, the energy consumption of a transportation system should be minimized. In general, the resulting layout has to be an adequate compromise of many parameters. This means that there are many possible ways to solve the problem, which quality differs.**

Consequently, this type of problem implies an explosion of search-space states with raising number of midpoints and/or resolution. In these cases, heuristic search methods have their major advances compared to non-heuristic search methods, which would need time and memory proportional to the search-space size to find the best solution. We explored, in particular, the use of a genetic algorithm, as a representative of the class of evolutionary algorithms. It searches the search-space selectively by focusing on interesting regions, constantly trying to find even better regions while having significantly less memory requirements. The importance of its main parameters, their impact on the performance and the precision of the genetic algorithm are presented in this paper. Guidelines for a good set-up of the genetic algorithm are given in order to gain high efficiency and effectiveness; the encoding of parameters, the build-up of the fitness function, evaluation and reproduction of chromosomes, elitism and convergence affinity as the main engine of genetic algorithms is discussed in detail as are the limits of this approach.

The modified and improved simple genetic algorithm to solve the mentioned track-layout problem forms another core part of the paper.



Motivation/General observation

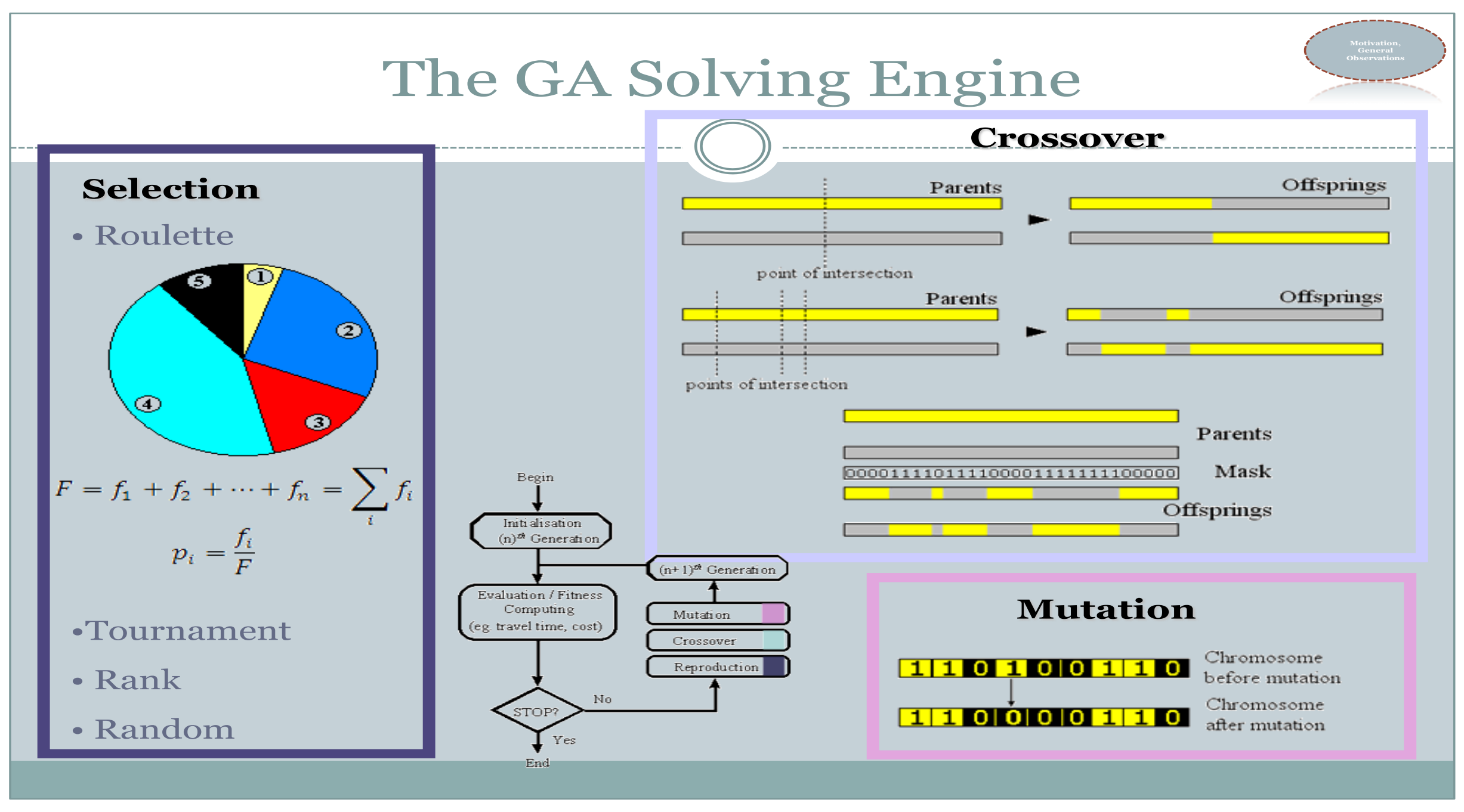
1. Idea: Track layout in a given area; plan!
2. Hint: The area could be a satellite scan
3. Task: Finding a compromise between
   - Track length: ↘
   - Summed elevation of the path: → 0
   - Choice of track length midpoints and resolution (because of serch space combinatorialexpolosion)

How to simulate and implement our idea, concept, model?

In order to minimize the energy consumption of transportation vehicles, and having in mind imminent energy crisis, we have considered: 1, 2, 3

We can define the genetic algorithm method as 'a probabilistic search algorithm that iteratively transforms a set (called a *population*) of mathematical objects (typically fixed-length binary character strings), each with an associated fitness value, into a new population of offspring objects using the Darwinian principle of natural selection and using operations that are patterned after naturally occurring genetic operations, such as crossover (sexual recombination) and mutation'.

In accordance with this, the analogy is as following: our mathematical object (chromosome – fixed lenght binary character string) represents the coordinates of the points which delimit the subsegments of the path – the actual track layout (or one possible combination, to be more precise). A set (population) of these mathematical objects forms a generation. The next generation is formed as offsprings of the actual population, by recombining them (crossover) with a probability proportional to their fitness and randomly flipping some bits (mutation).



The GA Solving Engine

Selection
- Roulette

$$F = f_1 + f_2 + \cdots + f_n = \sum_i f_i$$

$$p_i = \frac{f_i}{F}$$

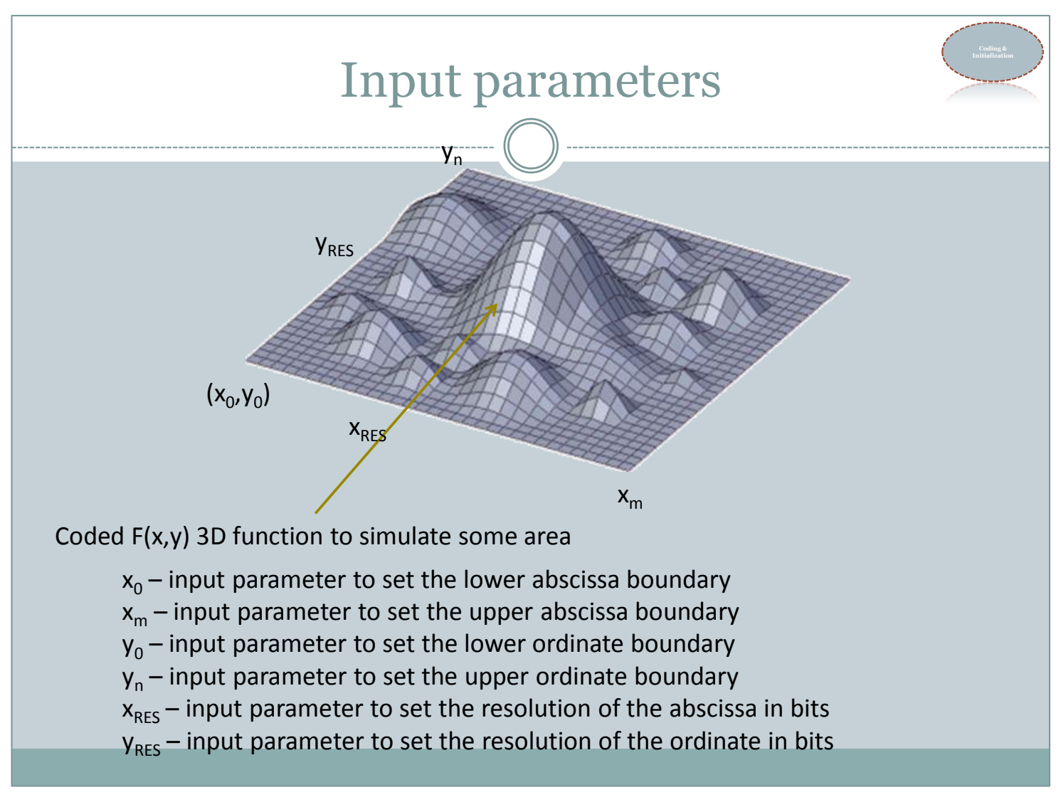
- Tournament
- Rank
- Random

Crossover

Mutation

CODING AND INITIALIZATION

The track layout problem has some specific implementation wattles. Our chromosome represents the path as a feasible solution to the problem, a set of coupled points binary coded by the (x,y) coordinates of the path points. The geographical area (see Fig.), which we search the solution upon, is delimited by the lower (x_0) and upper (x_m) abscissa boundary and the lower (y_0) and upper (y_n) ordinata boundary – the path onset is fixed at the point (x_0, y_0) and the end is fixed at (x_m, y_n); these points are not being coded into the chromosome, because their positions will never change.



Input parameters

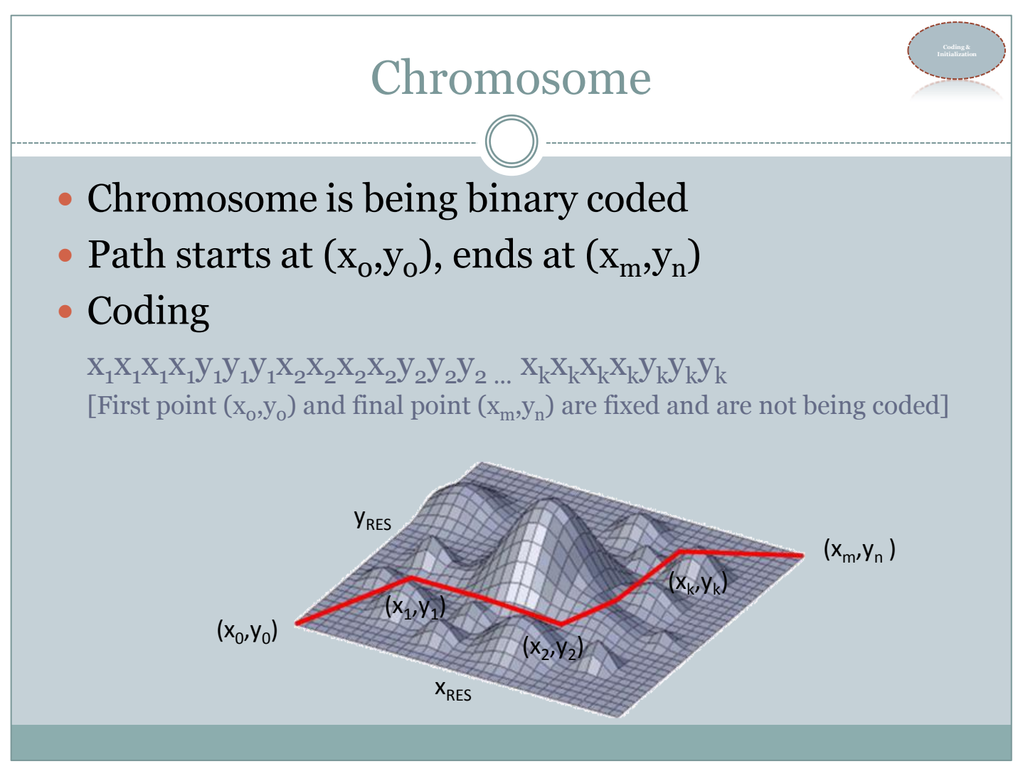
Coded F(x,y) 3D function to simulate some area

- x_0 – input parameter to set the lower abscissa boundary
- x_m – input parameter to set the upper abscissa boundary
- y_0 – input parameter to set the lower ordinate boundary
- y_n – input parameter to set the upper ordinate boundary
- x_{RES} – input parameter to set the resolution of the abscissa in bits
- y_{RES} – input parameter to set the resolution of the ordinate in bits

The parameters $x_0, x_m, y_0, y_n, k, x_{RES}, y_{RES}$, where k represents the number of midpoints and x_{RES}, y_{RES} the abscissa and ordinate resolution in bits respectively, are variable input parameters. The residual coordinates form the chromosome as an array of bits, which code the midpoint coordinates in standard binary coding at the chosen resolution.

$$x_1x_1x_1x_1y_1y_1y_1 \ldots x_kx_kx_kx_ky_ky_ky_k$$

Every point (except the start and the endpoint) in this model is coupled directly to its neighbor, which means that P_i with the coordinates (x_i, y_i) is coupled with $P_{i-1}: (x_{i-1}, y_{i-1})$ (as its predecessor) and $P_{i+1}: (x_{i+1}, y_{i+1})$ (as its successor), while $0 < i < k, i \in \mathbb{N}$



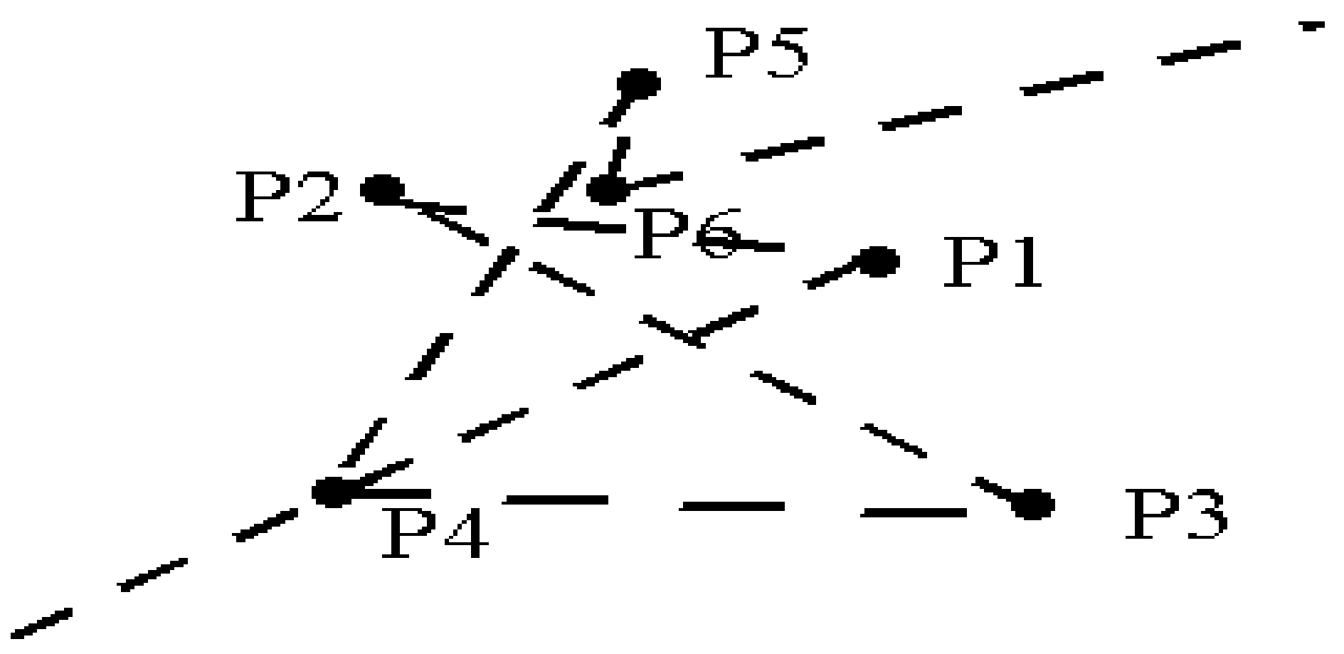
Chromosome

- Chromosome is being binary coded
- Path starts at (x_o,y_o), ends at (x_m,y_n)
- Coding

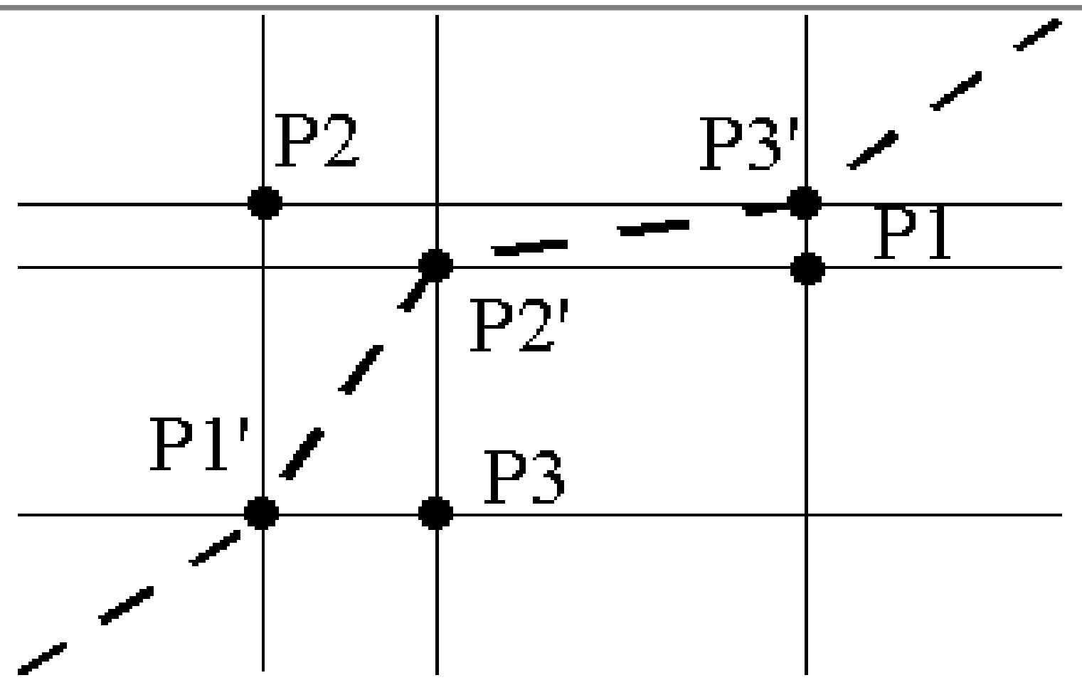
$x_1x_1x_1x_1y_1y_1y_1x_2x_2x_2x_2y_2y_2y_2 \ldots x_kx_kx_kx_ky_ky_ky_k$

[First point (x_o,y_o) and final point (x_m,y_n) are fixed and are not being coded]

There are many ways to avoid the situation shown in Figure. One possible way is to generate the coordinates randomly, but by discarding and regenerating them if they are lower (by their x and y parts) of the predecessor point coordinates.

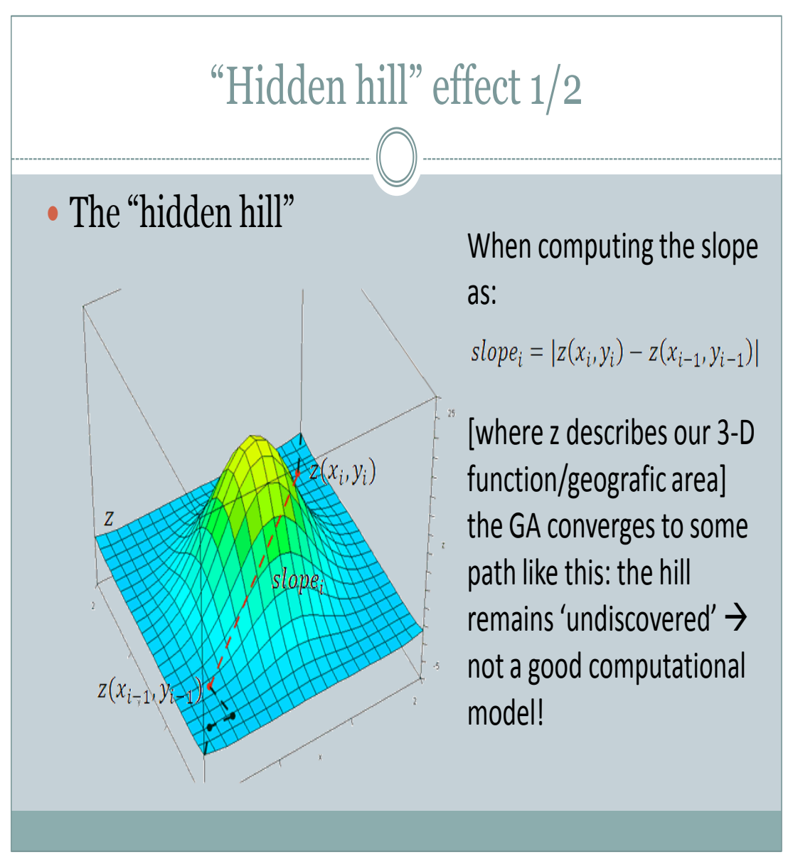


The negative effect of this approach is the number of increasing loops in the initialization process. Another feasible solution to the initialization problem is the creation of a path, which is formed by ascending sort of randomly generated x and y coordinates.



The genetic algorithm works much better when having a population of paths without loops and drawbacks.The effectiveness is being increased significantly, the convergence ability is improved.

The genetic algorithm is able to set the lengths of the path segments as they fit better into the fitness function; the result: the GA creates a path which is in fact formed of points with minimal heigth difference – yet, the segments have in real a big slope.



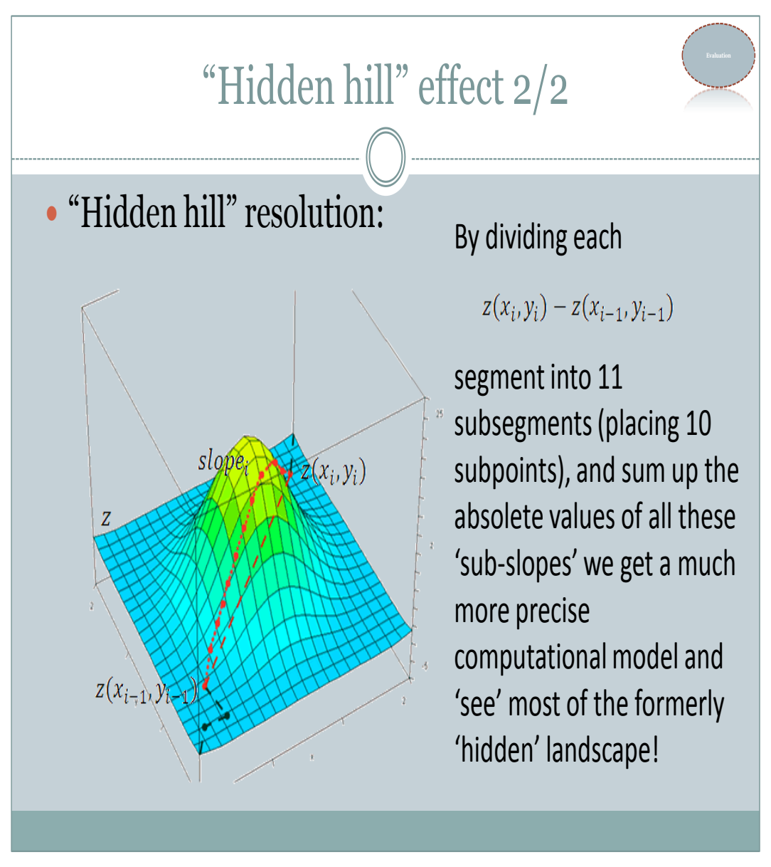
"Hidden hill" effect 1/2

- The "hidden hill"

When computing the slope as:

$$slope_i = |z(x_i,y_i) - z(x_{i-1},y_{i-1})|$$

[where z describes our 3-D function/geografic area] the GA converges to some path like this: the hill remains 'undiscovered' → not a good computational model!



"Hidden hill" effect 2/2

- "Hidden hill" resolution:

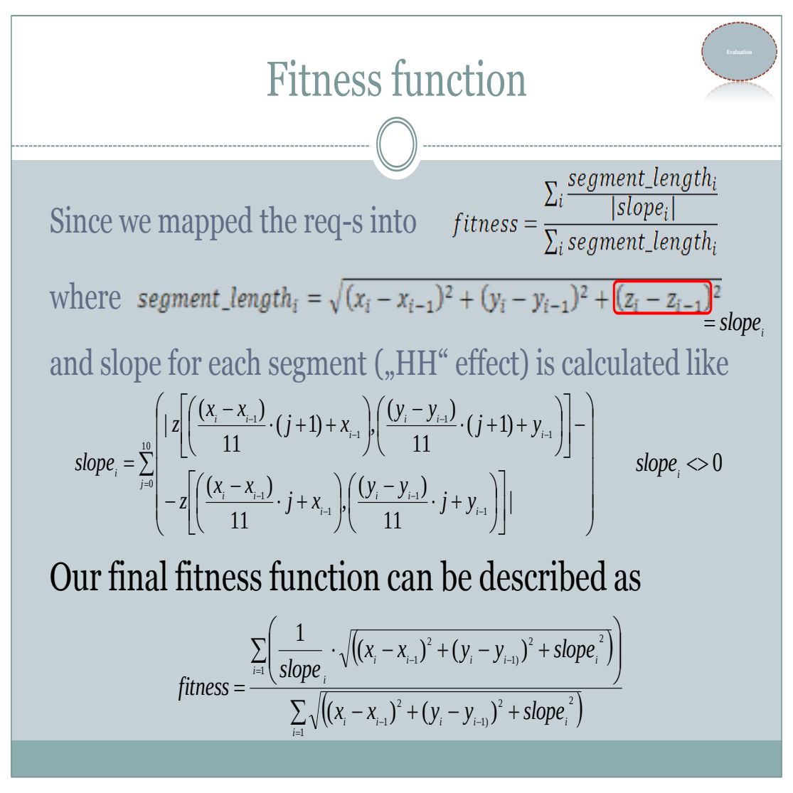
By dividing each

$$z(x_i,y_i) - z(x_{i-1},y_{i-1})$$

segment into 11 subsegments (placing 10 subpoints), and sum up the absolete values of all these 'sub-slopes' we get a much more precise computational model and 'see' most of the formerly 'hidden' landscape!

The resolution is to increase the number of midpoints, which means to set additional points between on every path segment, crating a more precise computational model. If we set the number of subpoints on each segment to 10, we get 11 subsegments for every path segment.

The precision of the fitness function directly depends on the used resolution: the higher the resolution is, the merely precise is our slope approximation (and by that also the fitness calculation). We have also to mention, that the calculation time is raised while increasing the precision of the calculation. In some cases it would be well advised to start the genetic search with a lower resolution and raise it by narrowing the search area. The result of the advanced slope calculation used by the fitness function is shown on the next Figure.



Fitness function

Since we mapped the req-s into

$$fitness = \frac{\sum_i \frac{segment\_length_i}{|slope_i|}}{\sum_i segment\_length_i}$$

where

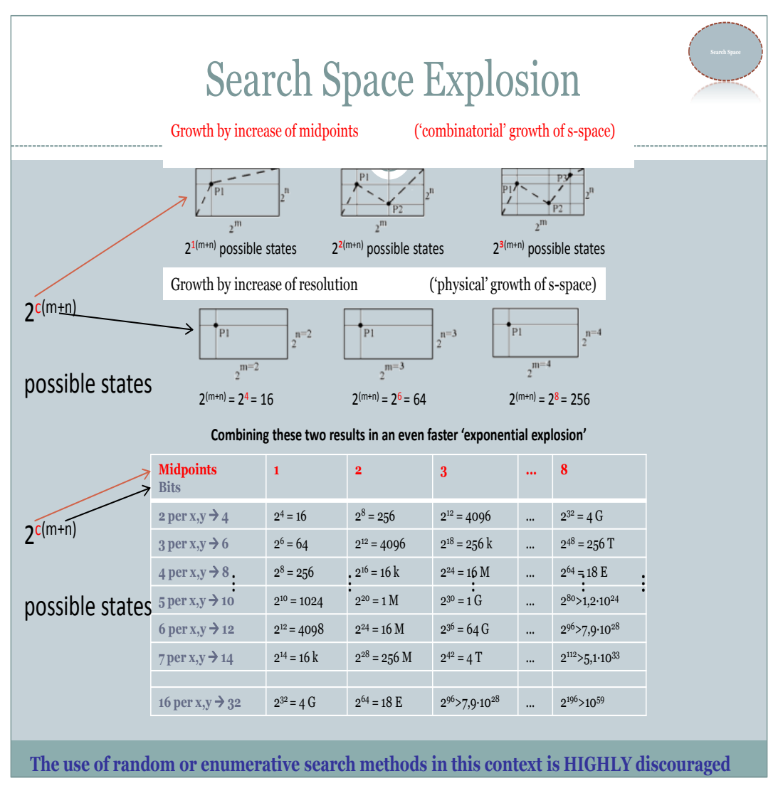
$$segment\_length_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + \boxed{(z_i - z_{i-1})}^2}, \quad (z_i - z_{i-1}) = slope_i$$

and slope for each segment („HH" effect) is calculated like

$$slope_i = \sum_{j=0}^{10} \left(\left| z\left[\left(\frac{(x_i - x_{i-1})}{11}\cdot(j+1) + x_{i-1}\right), \left(\frac{(y_i - y_{i-1})}{11}\cdot(j+1) + y_{i-1}\right)\right] - z\left[\left(\frac{(x_i - x_{i-1})}{11}\cdot j + x_{i-1}\right), \left(\frac{(y_i - y_{i-1})}{11}\cdot j + y_{i-1}\right)\right] \right| \right) \qquad slope_i <> 0$$

Our final fitness function can be described as

$$fitness = \frac{\sum_{i=1}\left(\frac{1}{slope_i}\cdot\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + slope_i^2}\right)}{\sum_{i=1}\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + slope_i^2}}$$



Search Space Explosion

Growth by increase of midpoints ('combinatorial' growth of s-space)

P1
P2
2^m
2^n
$2^{1(m+n)}$ possible states
$2^{2(m+n)}$ possible states
$2^{3(m+n)}$ possible states

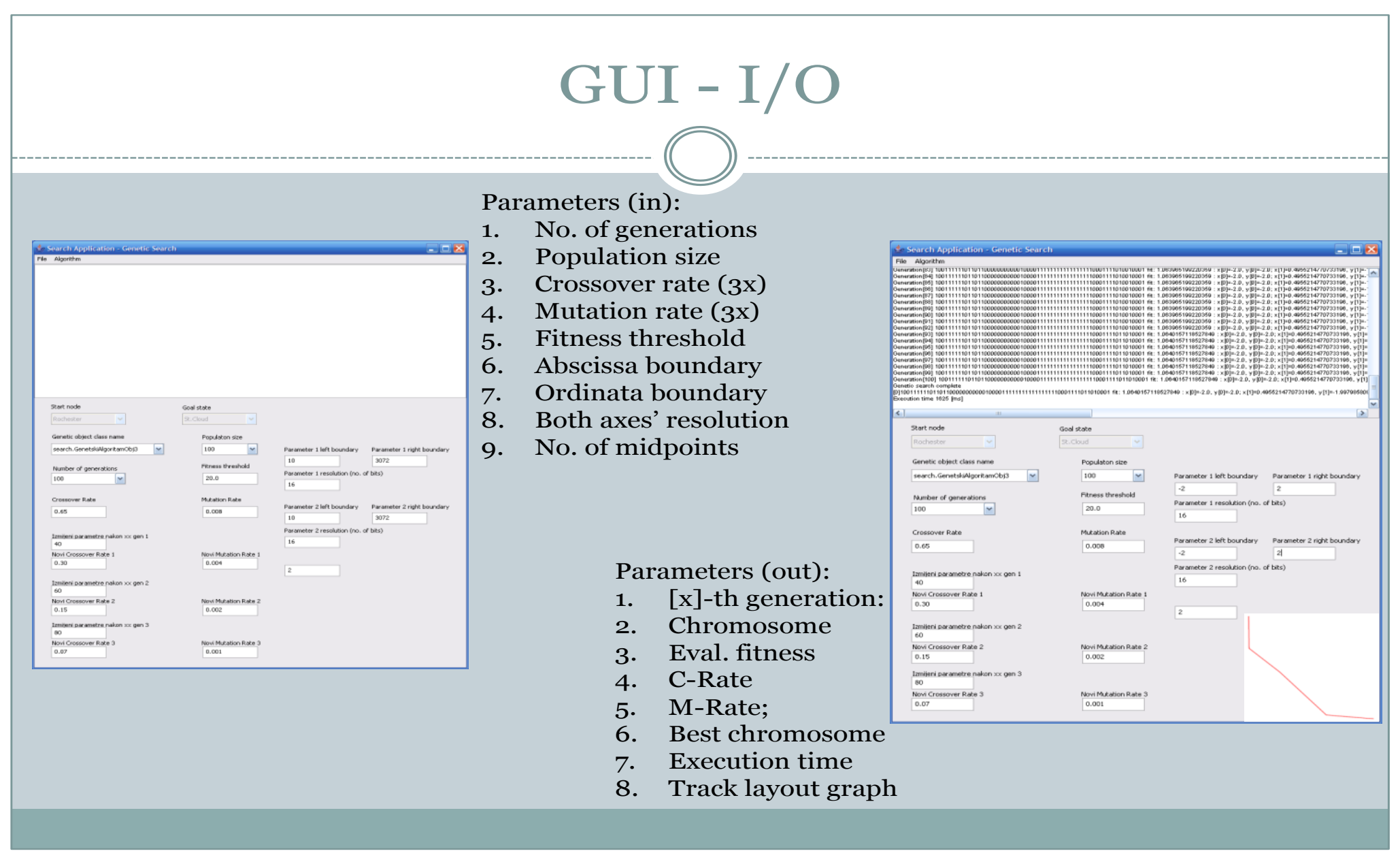
Growth by increase of resolution ('physical' growth of s-space)

Combining these two results in an even faster 'exponential explosion'

$2^{c(m+n)}$ possible states

Midpoints Bits	1	2	3	...	8
2 per x,y → 4	$2^4 = 16$	$2^8 = 256$	$2^{12} = 4096$	...	$2^{32} = 4$ G
3 per x,y → 6	$2^6 = 64$	$2^{12} = 4096$	$2^{18} = 256$ k	...	$2^{48} = 256$ T
4 per x,y → 8	$2^8 = 256$	$2^{16} = 16$ k	$2^{24} = 16$ M	...	$2^{64} = 18$ E
5 per x,y → 10	$2^{10} = 1024$	$2^{20} = 1$ M	$2^{30} = 1$ G	...	$2^{80} > 1{,}2\cdot10^{24}$
6 per x,y → 12	$2^{12} = 4098$	$2^{24} = 16$ M	$2^{36} = 64$ G	...	$2^{96} > 7{,}9\cdot10^{28}$
7 per x,y → 14	$2^{14} = 16$ k	$2^{28} = 256$ M	$2^{42} = 4$ T	...	$2^{112} > 5{,}1\cdot10^{33}$
16 per x,y → 32	$2^{32} = 4$ G	$2^{64} = 18$ E	$2^{96} > 7{,}9\cdot10^{28}$	...	$2^{196} > 10^{59}$

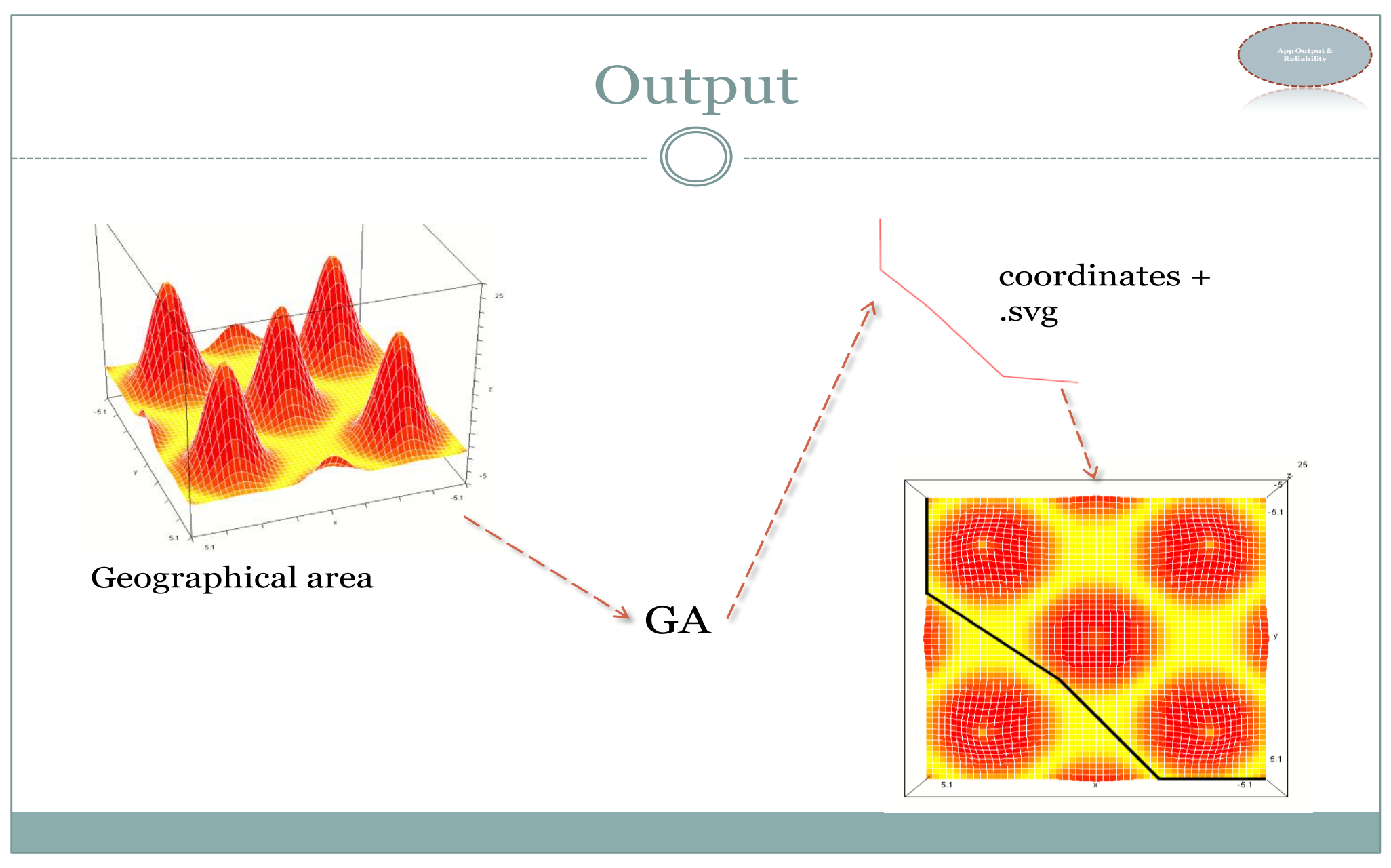
The use of random or enumerative search methods in this context is HIGHLY discouraged



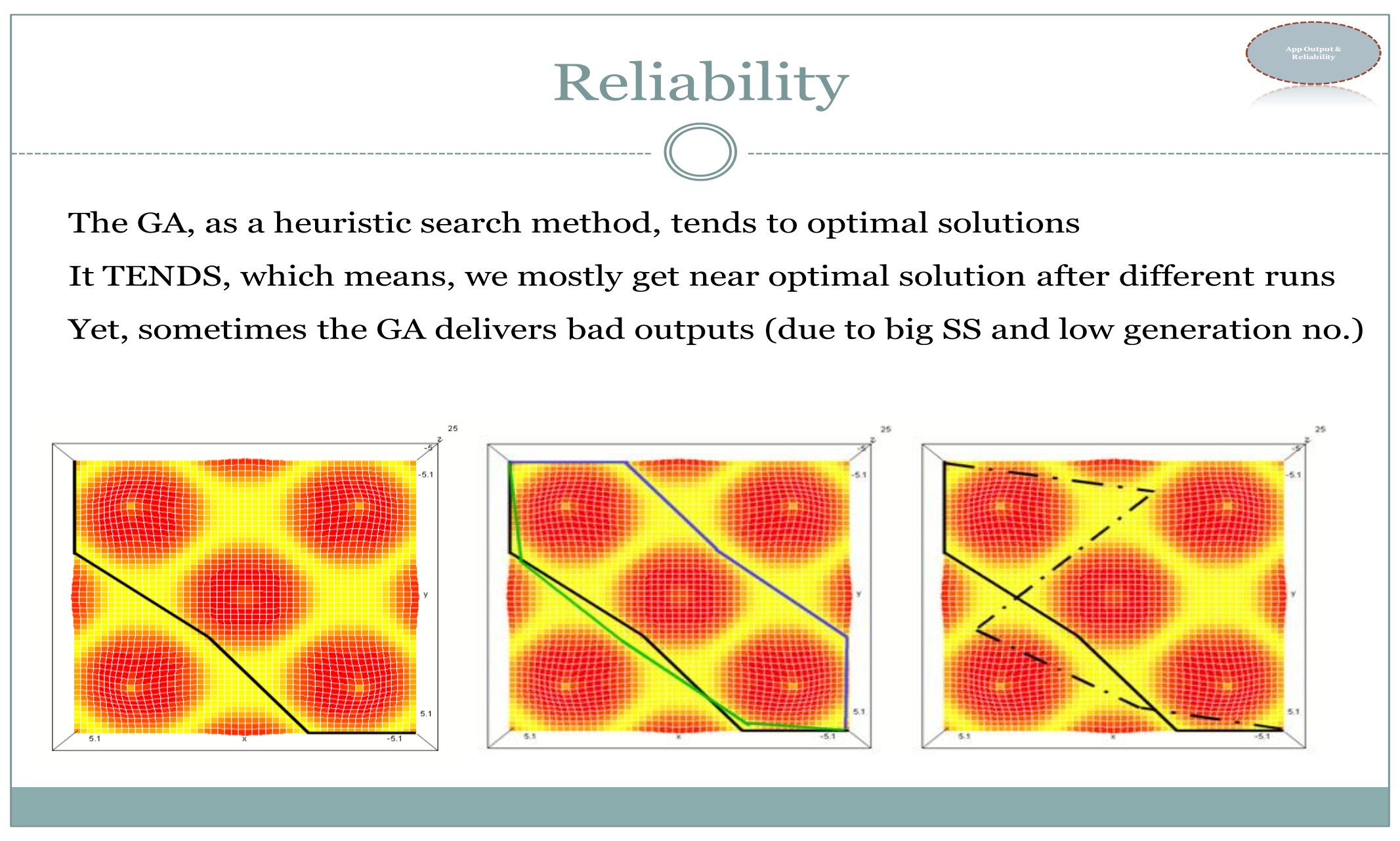
GUI - I/O

Parameters (in):
1. No. of generations
2. Population size
3. Crossover rate (3x)
4. Mutation rate (3x)
5. Fitness threshold
6. Abscissa boundary
7. Ordinata boundary
8. Both axes' resolution
9. No. of midpoints

Parameters (out):
1. [x]-th generation:
2. Chromosome
3. Eval. fitness
4. C-Rate
5. M-Rate;
6. Best chromosome
7. Execution time
8. Track layout graph



Output



Reliability

The GA, as a heuristic search method, tends to optimal solutions

It TENDS, which means, we mostly get near optimal solution after different runs

Yet, sometimes the GA delivers bad outputs (due to big SS and low generation no.)

CONCLUSION

The genetic algorithm provides fast, precise and good results until around 4 midpoints. Beyond these, the imprecision raises – due to the complexity and linkage of midpoints. The fitness function evaluates the fitness of the whole chromosome, which means of the whole path, and not the path segments. If we have a path, which is well-designed over its 90% of length, but one point bails out (and this could be just 1 bit of hundreds), its fitness is ruined, and it might not 'survive'. The fitness function should be revisited in order to try to solve this problem, and the genetic operators could be modified. We could also try out to implement the Gray code instead of the regular binary code in order to make small modifications to the path when mutating it (the binary code makes combined - small, medium and big – modifications to the chromosome, depending on the 'importance' of the changed bit.

Another interesting challenge is the application of this algorithm on real geographical areas, which could be coded into some data arrays from a satellite scan. This array could even be a structure of different parameters, like the height, the soil condition and other relevant parameters for track or road layout. These parameters could then be additional input parameters for the fitness function which would help to design the problem even more realistic.

LITERATURE

[1] J.P. Bigus, J. Bigus., *Constructing Intelligent Agents Using Java, New York: John Wiley & Sons, Inc., 2001.*
[2] Z. Avdagic, *Artificial Intelligence & Expert Systems, Sarajevo: Grafoart, 2003.*
[3] Z. Avdagic, *Artificial Intelligence & Fuzzy-Neuro-Genetic, Sarajevo: Grafoart, 2003.*
[4] G. E. Cantarella and A. Vitetta, "The multi-criteria road network design problem in an urban area" *Transportation, vol. 33, pp. 567-588, November 2006.*
[5] A. Nagar and H.Tawfik., "A Multi-Criteria Approach in Prototyping Urban Road Networks", in *Informing Science + Information Technology Education Joint Conference, Ljubljana, Slovenia, 2007.*
[6] Z. Avdagic, S. Konjicija, and F.Dervisbegovic, "Genetic Algorithms in Optimization of Engineering Problems" in *Proceedings- Information and Communication Technologies, XVIII Symposium, Sarajevo, 2001.*
[7] B. Hillier, *Space is the machine Cambridge: Cambridge University Press, 1996.*
[8] R. Béra and C. Claramunt, "Can the relative adjacency contribute to space syntax in the search for a structural logic of the city", *International GI Science Conference, Washington, 2004.*
[9] J. Peponis, C. Zimring, and Y.K. Choi, "Finding the building in wayfinding" *Environment and Behavior, vol. 33, pp. 555-590, 1990.*
[10] P. Ortega and L. Jose, "Reference analysis base on a vectorial spaces model: contemporany history in Jaén research for 1990 – 1995" *Revista Española de Documentación Científica, vol. 24, 2001.*
[11] S. Porta, P. Cruciti, and V. Latora, "The network analysis of urban streets: A dual approach" *Physica A: Statistical and Theoretical Physics, vol. 369, pp. 853-866, 2006.*
[12] B. Jiang and C. Claramunt, "A Structural Approach to the Model Generalization of an Urban Street Network", *GeoInformatica, vol. 8, pp. 157-171, 2004.*